ششمین همایش بین‌المللی افق های نویـن در
مهندسی برق، کامپیوتـر و مکانیک
6th International Conference on the New Horizons in
Electrical Engineering, Computer and Mechanical
www.mhconf.ir

فعالیت های ضروری برای توسعه نرم افزار ایمن

محمدجوادحسین پور[۱،*]، صابرصادقی[۲]،مژگان زارعی[۳]، سعیده زردشت[۴]

[۱،*]عضو هیئت علمی و استادیار بخش مهندسی کامپیوتر،دانشگاه آزاد اسلامی واحد استهبان،استهبان،ایران

Email: mj.hosseinpoor@iau.ac.ir

[۲]دانشجوی کارشناسی ارشد مهندسی کامپیوتر، دانشگاه آزاد اسلامی واحد استهبان،استهبان،ایران

Email: ssabersadeghii@yahoo.com

[۳] دانشجوی کارشناسی ارشد مهندسی کامپیوتر، دانشگاه آزاد اسلامی واحد استهبان،استهبان،ایران

Email: mojhizare@gmail.com

[۴] دانشجوی کارشناسی ارشد مهندسی کامپیوتر، دانشگاه آزاد اسلامی واحد استهبان،استهبان،ایران

Email: saeedeh_z60@yahoo.com

چکیده

انواع مختلفی از نرم افزارها تقریباً در تمام بخش های تجارت در دنیای مدرن استفاده می شود. آنها مکانیسم هایی را ارائه می دهند که خریداران و فروشندگان را قادر می سازد تا به صورت مجازی با هم تعامل داشته باشند، کار دستی را در مشاغل و موسسات کاهش دهند و همچنین کار را بسیار آسان تر کنند. افزایش تقاضا برای نرم افزار منجر به افزایش سرمایه گذاری شده است که متعاقباً حملات امنیتی متعددی را به خود جلب کرده است. میلیون‌ها منبع در نرم‌افزارهای مختلف در سراسر جهان نگهداری می‌شوند، مجرمان حملات سایبری در نقض امنیت نرم‌افزار برای دستاوردهای خودخواهانه حرفه‌ای کرده‌اند، بنابراین توسعه و استقرار نرم‌افزار امن را ضروری می‌سازند. این کار از طریق بررسی ادبیات، مفاهیم و اصطلاحات مورد استفاده در توسعه نرم‌افزار ایمن را معرفی می‌کند، بهترین شیوه‌ها را ارائه می‌کند و مروری بر مدل‌های قابل استفاده ارائه می‌دهد. محرمانه بودن، یکپارچگی، در دسترس بودن و عدم انکار اصطلاحات نرم افزاری امنی هستند که به این معنی است که باید محرمانه، ایمن و در دسترس باشد و هر فعالیت انجام شده را ثبت کند. کار پیشنهادی از چندین روش برتر حمایت می‌کند، از جمله ایجاد یک محیط امن که دسترسی به بخش‌ها یا بخش‌های کلیدی سیستم را محدود می‌کند و علاوه بر کاهش سطح حمله یا به جای کاهش فرصت‌های موجود برای حمله سایبری. در رابطه با مهندسی نرم افزار، این مقاله توصیه می کند که الزامات سیستم باید قبل از ایجاد نرم افزار ایجاد شود. مهندسی اضافی باید پس از ارزیابی سیستم درست قبل از راه اندازی رسمی انجام شود. علاوه بر این، این مقاله اتخاذ استراتژی‌هایی را توصیه می‌کند که توسط مدل‌های نرم‌افزار معروفی مانند چرخه عمر توسعه نرم‌افزار مایکروسافت در میان دیگران استفاده می‌شوند. این مدل ها استراتژی های نرم افزاری ایمن را در طول چرخه عمر توسعه نرم افزار قرار داده اند. آنها نیاز به قرار دادن سیستم های مهندسی ایمن در طول طراحی و استفاده از نرم افزار را تشخیص می دهند زیرا هر روز روش های جدیدی برای نقض امنیت نرم افزار مطرح می شود. این مقاله با ذکر این نکته پایان می‌یابد که تلاش‌های مشترک مستمر برای تضمین نرم‌افزار امن‌تر هنوز یک نیاز ضروری است. پایبندی به توسعه و استفاده از نرم افزار ایمن اساسی علاوه بر توسعه مهندسی اضافی که یکپارچگی، محرمانه بودن و دسترسی به نرم افزار را حفظ می کند، ضروری است.

کلید واژه ها:مهندسی نرم افزار، کیفیت نرم افزار، توسعه امنیت نرم افزار.

۱. مقدمه

جامعه ما پر از فعالیت هایی است که یا با کمک یک نرم افزار یا درون نرم افزار انجام می شود. همه انواع کسب و کارها به نوعی از نرم افزار به گونه ای استفاده می کنند که انسان را به استفاده از نرم افزار معتاد کرده است [۶]. نگرانی امنیت تراکنش ها، اطلاعات شخصی و دارایی های ارزشمندی است که در آن سیستم های نرم افزاری نگهداری می شود. از این رو نیاز انسان به ارزیابی ایمنی سیستم های نرم افزاری است. با این حال، امنیت سیستم‌های نرم‌افزاری می‌تواند یک موضوع کاملاً گسترده باشد. اما اخبار حملات سایبری از دست رفته و قطع خدمات در موسسات دولتی و غیردولتی رایج است. ماهیت افسارگسیخته حملات نرم افزاری موفق، روحیه بسیاری از کسانی را که امیدوار به سیستم های امن تر هستند، تضعیف کرده است.

مجله بین المللی مهندسی نرم افزار و برنامه های کاربردی

با وجود ماهیت پیچیده آن، امنیت سیستم های نرم افزاری را می توان با پایبندی به اصول اولیه ای که خطرات را به حداقل می رساند، افزایش داد [۶، ۱۴]. علاوه بر این، ذینفعان باید بدانند که امنیت نرم افزار یک هدف نیست، بلکه اگر یک سفر مداوم است که مستلزم توجه است و سریعتر است که نه از حرکت می کند. همچنین به این معنی است که خطرات را نمی توان به طور کامل حذف کرد، اما می توان آن را بدون آسیب رساندن به افراد و مؤسسات با استفاده از یک نرم افزار خاص به کمترین کاهش داد.

برای خطر باید بر اساس راه های احتمالی وجود داشته باشد که می توان در استفاده از آن [۵]. محاسبه الزامی است زیرا چگونه یا زمان انجام یک حمله خاص را نشان می دهد [۳]. همچنین کسب دانش و مهارت برای جلوگیری و کاهش آن هدف از طریق به موقع و مناسب با میزان خطر پیش بینی شده را امکان پذیر می کند. اصطلاحات با توجه به امنیت نرم افزار استفاده می شود [۱۸]. محرمانه مستلزم این است که سیستم اطلاعات را مخفی نگه می دارد، یکپارچگی این است که سیستم مستلزم آن است که مستلزم مستلزم آن است که به طور قانونی در دسترسی و استفاده از کاربران خود را حفظ کند. سیستم های نرم افزاری ایمان باید الزامات یکپارچگی، دسترسی و محرمانگی را برآورده کنند [۱۸].

علاوه بر آن، سیستم باید سوابق و گزارش‌های مربوط به انواع فعالیت‌های مختلف را که از طریق آن اتفاق افتاده است، نگه می‌دارد. باید همه فعالیت‌ها از طریق آن‌ها انجام شود و آن‌ها را انجام دهند، می‌توانند انجام دهند، که این ویژگی معمولاً به عنوان عدم انکار از آن یاد می‌شود. امنیت نرم‌افزار مفهومی است که به معنی‌زار ارائه مهندسی برای اطمینان از حفظ یکپارچگی، محرمانه بودن، قابلیت دسترسی و ویژگی‌های عدم انکار نرم‌افزار برای عملکرد در میان عملیات‌های متعددی است که در روزگار مدرن انجام می‌شود مدل کلی چرخه عمر توسعه نرم افزار امن (SSDLC) و اجزای امنیتی را با مطابقت با فازها نشان می دهد.

۲. بهترین روش ها برای توسعه نرم افزار ایمن

امنیت نرم افزار دارای مفاهیم متعددی است [۱۳]. این مهم است که نرم افزارهایی که از سیستم های نرم افزاری استفاده می کنند و در حال توسعه آن هستند، از مفاهیم و اصول اساسی که یکپارچگی، دسترسی و محرمانه بودن سیستم های نرم افزاری خود را بهتر می دهد، هستند. برخی از مفاهیم مانند محیط امنیتی و سطح حمله [۱۰] از جمله در زیر ارائه شده است. محیط امنیتی یک اصطلاح جدید بلکه یک رویه رایج

است. ساختمان ها با استفاده از دیوار محیطی امنیتی به عنوان اولین خط دفاعی در برابر تهدیدات خارجی ایمن می شوند. این مفهوم در همه تاسیسات کلیدی

مانند فرودگاه ها، پادگان های نظامی و مدارس در میان سایر موارد به عنوان اولویت در ایجاد یک منطقه امن در داخل دیوار محیطی تکرار می شود. یک واحد ورودی ایجاد می‌شود به طوری که همه کسانی که به منطقه امن دسترسی دارند می‌توانند به اندازه کافی قابل دسترسی باشند، می‌خواهند آنها را مشخص کنند و یکپارچگی منطقه امن مداخله نمی‌کنند. در توسعه نرم افزار امنیت، ایجاد یک محیط امنیتی برای محافظت از بخش های کلیدی نرم افزار در برابر دسترسی های مجاز و مخرب بسیار مهم است. با این حال، ماهیت دائماً در حال تغییر تجارت، شبکه و افزایش گشودگی به سهامداران جهانی و مشتریان، تعیین مکان های امنیتی محیطی را به چالش کشیده است.

سطح حمله مفهوم دیگری است که ممکن است به منظور افزایش امنیت سیستم های نرم افزاری به کار رود. این تا حد زیادی در ارزیابی خطر و ایجاد اقداماتی با هدف نه تنها از بین بردن خطر، بلکه همچنین کاهش اثر ناشی از یک حمله موفقیت آمیز استفاده می شود. سطح حمله تمام نقاط ورود امکان پذیر را تعریف می کند که یک فرد مخرب ممکن است از آنها برای دسترسی به منطقه امنیتی یک نرم افزار استفاده کند. این به معنای کل مناطق حمله احتمالی است، صرف نظر از اینکه مهاجم از محیط بیرونی کار می کند یا از محیط داخلی. این مفهوم کاهش سطح حمله را به حداقل ممکن توصیه می کند زیرا هر چه تعداد نقاط ورودی بیشتر باشد، تضمین یکپارچگی سیستم نرم افزار دشوارتر است. علاوه بر این، توسعه نرم‌افزار ایمن بیش از طراحی، بلکه به کاربرد رفتارهای اخلاقی در توسعه، نصب و استفاده از سیستم‌های نرم‌افزاری نیاز دارد. اصول مدیریت نرم افزار از این جهت مفید هستند که به تصمیم گیری هایی کمک می کنند که امنیت سیستم را ارتقاء دهند. با بکارگیری اصول سیستم نرم افزار ایمن، الزامات سیستم به خوبی برقرار می شود، طراحی با تمرکز بر کاهش سطح حمله انجام می شود و سیستم به گونه ای پیاده سازی می شود که پیشرفت های جدیدی را که نیازمند به روز رسانی منظم سیستم هستند، مورد توجه قرار دهد.

پروژه امنیت برنامه وب باز 10 (OWASP) اصل را که در فهرست شده است مورد بحث قرار داده است:. اصول امنیت نرم افزار و بهترین روش ها

۲,۱. تمرین ۱ به کارگیری دفاع در عمق

این اصل بیان می‌کند که می‌توانید امنیت را با ساختار دادن به کنترل‌های امنیتی خود به‌عنوان دنباله‌ای از لایه‌های همپوشانی برای ارائه سه مورد ضروری برای ایمن کردن دارایی‌ها (پیشگیری، شناسایی و پاسخ) افزایش دهید. در توسعه نرم افزار، اقدامات متقابل و کنترل های امنیتی باید برای محافظت، شناسایی و پاسخ به حملات امنیتی احتمالی لایه بندی شوند. دفاع در عمق، انتخاب کنترل‌ها را برای یک برنامه راهنمایی می‌کند تا از انعطاف‌پذیری آن در برابر انواع مختلف حملات اطمینان حاصل کند و احتمال یک نقطه شکست را کاهش دهد.

۲,۲. تمرین ۲ از یک مدل امنیتی مثبت استفاده کنید

استفاده از یک مدل امنیتی مثبت - که لیست سفید نیز نامیده می شود - تعیین می کند که چه چیزی مجاز است و هر چیز دیگری را رد می کند. مدل موجود دیگر یک مدل امنیتی "منفی" (یا "لیست سیاه") است که تعیین می کند چه چیزی را می توان غیر مجاز دانست و هر چیز دیگری را مجاز دانست. یکی از مسائل در توسعه نرم افزار نیاز به "شمارش بد" یا شروع استفاده از لیست سیاه است که یک فرآیند بی نهایت است. در لیست سفید بر "شمارش خوبی ها" تمرکز می شود که کار ساده تر و کارآمدتری است. توسعه دهندگان می توانند فقط بر روی قالب های قابل قبول تمرکز کنند و هر چیز دیگری را که منجر به جلوگیری از حملات جدید - از جمله حملات روز صفر - می شود که توسط توسعه دهندگان پیش بینی نشده است، رد کنند.

۲,۳. ۳ شکست را ایمن تمرین کنید

یکی از جنبه های کلیدی برنامه های کاربردی ایمن و انعطاف پذیر، مدیریت ایمن خطاها و استثناها است. دو نوع اصلی از خطاها نیاز به مراقبت ویژه دارند، یعنی استثناهایی که هنگام پردازش یک کنترل امنیتی رخ می‌دهند و استثناهای کدی که «مرتبط با امنیت» نیستند. مهمترین چیز این است که این خطاها و استثناها اجازه رفتاری را نمی دهند که اقدامات متقابل امنیتی معمولاً اجازه نمی دهد. توسعه دهندگان باید بدانند که همیشه سه نتیجه ممکن از هر اقدام متقابل امنیتی وجود دارد: اجازه دادن، غیرمجاز کردن عملیات یا استثنا. در صورت خرابی در مکانیزم امنیتی،

مسیر اجرا باید همیشه به صورت پیش‌فرض به سمت غیرمجاز کردن عملیات برود. نوع دیگری از استثناهای مربوط به امنیت، مربوط به امنیت است، اگر بر اینکه برنامه به درستی کنترل امنیتی را فراخوانی کند یا خیر، تأثیر بگذارد. یک استثنا ممکن است باعث شود که یک روش امنیتی در زمانی که باید فراخوانی نشود، یا ممکن است روی مقداردهی اولیه متغیرهای مورد استفاده در کنترل امنیتی تأثیر بگذارد.

۲,۴. ۴ را با کمترین امتیاز اجرا کنید

این اصل بیان می کند که کاربران باید همیشه حداقل امتیازاتی را داشته باشند که به آنها اجازه می دهد نیازهای تجاری اولیه خود را برآورده کنند. این اصل معمولاً به عنوان یک تصمیم مهم طراحی پذیرفته شده است که محافظت از داده ها و عملکرد را در برابر خطاها و رفتارهای مخرب افزایش می دهد. اصل کمترین امتیاز گاهی نام دیگری دارد که اصل حداقل اختیار ((POLAاست.

۲,۵. تمرین ۵ اجتناب از امنیت توسط مبهم

امنیت از طریق مبهم تلاش برای دستیابی به امنیت یک سیستم نرم افزاری با تکیه بر مشکل در یافتن یا درک روش ها، مکانیسم ها و کنترل های امنیتی است. تکیه بر محرمانه بودن اجرای یک سیستم یا کنترل ها، یک کنترل امنیتی ضعیف تلقی می شود که همیشه با شکست مواجه خواهد شد. تأثیر روانی حفظ محرمانگی الگوریتم های پیاده سازی شده منجر به عدم موفقیت در اجرای آنها می شود. تنها اتکا به رازداری ممکن است به آسیب‌پذیری‌های امنیتی نظری یا واقعی منجر شود. امنیت باید با امنیت با طراحی به دست آید، نه با پنهان کاری. ۲,۶. تمرین ۶ امنیت را ساده نگه دارید

ساده نگه داشتن امنیت به جای پیچیده کردن چیزها، به کار بردن چیزهای ساده است. معماری، کد، کنترل امنیتی یا هر مصنوع باید نسبتاً ساده و ساده باشد تا کسی بتواند آن را بخواند و بفهمد. توسعه دهندگان باید همیشه برای رسیدن به سادگی و اجتناب از پیچیدگی تلاش کنند. ساده نگه داشتن امنیت با تعدادی از اصول انعطاف پذیری دیگر مرتبط است و استفاده از آن به عنوان یک اصل یا دستورالعمل به شما کمک می کند تا روح چندین اصل دیگر را برآورده کنید. یکی از راه های ساده نگه داشتن امنیت این است که توابع امنیتی را به این اهداف مجزا تقسیم کنیم:

- به طور پیش‌فرض، سرویس‌ها و اطلاعات مربوط به رد دسترسی را از مهاجمان دور نگه دارید.
- به کاربران مناسب اجازه دسترسی به اطلاعات مناسب مربوط به کمترین امتیاز را بدهید.
- از هر لایه به گونه ای دفاع کنید که گویی آخرین لایه دفاعی مرتبط با دفاع در عمق است.
- ثبت تمام تلاش ها برای دسترسی به اطلاعات (ورود به سیستم).
- تقسیم و جداسازی منابع.

۲,۷. تمرین ۷ تشخیص نفوذ

معمولاً، مهندسان نرم افزار می توانند با مطالعه ورودی های گزارشی که در زمان اجرا نمی توانید آنها را کشف کنید، مشکل نرم افزار را کشف کنند. تشخیص نفوذ در نرم‌افزار حداقل به سه عنصر اصلی نیاز دارد: قابلیت ثبت رویدادهای مرتبط با امنیت، رویه‌هایی برای اطمینان از نظارت منظم لاگ‌ها، و روش‌هایی برای پاسخ‌دهی مناسب به نفوذ پس از شناسایی. به ویژه، هرگونه استفاده از کنترل های امنیتی باید با اطلاعات دقیق کافی برای کمک به ردیابی مهاجم ثبت شود. از آنجایی که نمی‌توانید گزارش‌های رویداد را بررسی کنید تا نتیجه بگیرید کدام رویدادها قابل اجرا هستند، پس ثبت رویدادها ارزش افزوده‌ای ندارند. ورود به سیستم می تواند به عنوان یک عملکرد قانونی برای نرم افزار شما دیده شود. تشخیص نفوذ زمان مهاجم را برای تکمیل حملات خود محدود می کند. اگر نفوذها را به خوبی تشخیص دهید، مهاجم فقط یک بار تلاش می کند قبل از اینکه شناسایی شود و از انجام حملات بیشتر جلوگیری شود.

۲,۸. تمرین ۸ به زیرساخت ها اعتماد نکنید

شما هرگز نمی دانید که برنامه های شما در چه سخت افزاری یا محیط عاملی اجرا می شوند. تکیه بر فرآیند یا عملکرد امنیتی که ممکن است وجود داشته باشد یا نباشد، راه مطمئنی برای داشتن مشکلات امنیتی است. اطمینان حاصل کنید که الزامات امنیتی برنامه شما به صراحت از

طریق کد برنامه یا از طریق فراخوانی صریح توابع امنیتی قابل استفاده مجدد ارائه شده به توسعه دهندگان برنامه برای استفاده برای شرکت به عنوان مثال، (OWASP Enterprise Security API)ارائه شده است.

۲,۹. تمرین ۹ به خدمات اعتماد نکنید

خدمات می توانند به هر سیستم خارجی اشاره کنند. بسیاری از کسب‌وکارها از قدرت‌های پردازش شخص ثالث استفاده می‌کنند که به‌طور پیش‌فرض سیاست‌های امنیتی متفاوتی را اعمال می‌کنند که تحت این مشاغل نیستند. بنابراین، اعتماد ضمنی سیستم های اجرا شده خارجی تضمین نمی شود. همه سیستم های خارجی باید به روشی مشابه رفتار شوند.

۲,۱۰. تمرین ۱۰ ایجاد پیش فرض های ایمن

هر برنامه ای باید به طور پیش فرض خارج از جعبه به صورت ایمن تحویل داده شود! شما باید تصمیم گیری در مورد اینکه آیا می توانند امنیت خود را در صورتی که برنامه شما اجازه می دهد کاهش دهند، به عهده کاربران بگذارید. Secure به طور پیش‌فرض به این معنی است که تنظیمات پیکربندی پیش‌فرض ایمن‌ترین تنظیمات ممکن است – نه لزوماً کاربرپسندترین تنظیمات ممکن.

۳. مهندسی الزامات امنیتی

انتظار می‌رود که یک سیستم نرم‌افزاری دارای ویژگی‌های کلیدی باشد، مانند انجام کاری که قرار است انجام دهد، تعیین اینکه چه کسی با آن تعامل دارد و باید در میان سایر الزامات، بسیار خاص دامنه باشد. بدیهی است که یک نیاز سیستم را می توان به عنوان یک نیاز غیر کاربردی و/یا عملکردی طبقه بندی کرد. الزامات امنیتی سیستمی را که عملکردی و غیر کاربردی است متمایز می کند. یک سیستم امن باید در میان حملات مخرب خدماتی را بدون از دست دادن یکپارچگی و عملکرد خود ارائه دهد. بنابراین طراحی نرم افزار باید بر اساس الزامات امنیتی ایجاد شده باشد.

فرآیند تجزیه و تحلیل سیستم مورد نیاز نیز مهم است. ذینفعان باید در فرآیند شناسایی نیازمندی های سیستم مشارکت داشته باشند. زمان کافی نیز مهم است، برای اطمینان از اینکه تجزیه و تحلیل عجولانه با خطا انجام نمی شود. علاوه بر این، تجزیه و تحلیل باید در کل سیستم انجام شود و باید در اظهاراتی مستند شود که توصیف مناسب و دقیقی از سیستم ارائه دهد. یک SMART یادگاری (ویژه، قابل اندازه گیری، قابل دستیابی، واقع بینانه، قابل ردیابی) [۱۶] یک رویکرد ساده برای اطمینان از مستندسازی آن الزامات به اندازه کافی است. علاوه بر آن، تجزیه و تحلیل باید موارد استفاده و سوء استفاده/سوءاستفاده [۱۷] را ارائه دهد که ممکن است ایجاد شود تا اطمینان حاصل شود که طراحی تمام راه‌های حمله احتمالی را در نظر می‌گیرد. باید شبیه‌سازی شود که چگونه سیستم می‌تواند در طول یک مورد استفاده و همچنین در طول یک مورد سوء استفاده پاسخ دهد. به طور خلاصه، فرآیند مهندسی الزامات امنیتی را می توان به صورت گام به گام انجام داد. اولین گام، داشتن دانش قبلی از سطوح حمله در سیستم های دیگر مشابه با سیستم مورد استفاده است. پس از آن، آگاهی مهاجم را از نقاط ورود احتمالی ایجاد کنید و شبیه سازی کنید که چگونه مهاجم بر نقاط ضعف سیستم اعمال نفوذ می کند. یک شبیه سازی مشابه باید در مورد استفاده انجام شود تا مهندسی نیازمندی سیستم بتواند تمام شکاف ها را پر کند. به منظور استخراج الزامات امنیتی کارآمد، در اینجا چند مرحله توصیه می شود که باید دنبال کنید:

• با مطالعه مسائل امنیتی رایج برای سیستم هایی که مشابه سیستم شما هستند، با انجام تکالیف خود شروع کنید و ببینید آیا یک مهاجم ممکن است از چنین مسائلی در سیستم در حال توسعه سوء استفاده کند یا خیر. پس از آن، سعی کنید توضیح دهید که چگونه مهاجم در صورت وجود مشکل، از آن استفاده خواهد کرد.

• طوفان فکری بر اساس فهرستی از منابع سیستم. برای هر منبع، سعی کنید موارد سوء استفاده را در ارتباط با هر یک از سرویس های امنیتی اساسی ایجاد کنید: احراز هویت، محرمانه بودن، کنترل دسترسی، یکپارچگی و در دسترس بودن.

• طوفان فکری بر اساس مجموعه ای از موارد استفاده موجود. این ممکن است برای شناسایی ریسک های نماینده و برای اطمینان از اینکه دو رویکرد دیگر هیچ تهدید آشکاری را نادیده نمی گیرند مفید باشد. موارد سوء استفاده که به این روش به دست می‌آیند، اغلب بر حسب استفاده معتبر نوشته می‌شوند و سپس دارای مراحل مخرب هستند.

الزامات امنیتی باید مراحل مختلفی را طی کند، از استخراج تا تحلیل و مدل‌سازی و سپس اولویت‌بندی و مستندسازی. کارشناسان امنیتی، تحلیلگران تجاری، طراحان، معماران، توسعه‌دهندگان، آزمایش‌کنندگان، تیم‌های تعمیر و نگهداری همگی باید در جمع‌آوری و تجزیه و تحلیل

الزامات امنیتی مشارکت داشته باشند. چالش‌های ناشی از فرآیند ارزیابی نیازمندی‌های سیستم شامل هزینه‌ها، تهدیدات نوظهور و عدم درک تهدیدها در طول فرآیند طراحی است. فقدان چارچوب سیاسی قانونی و کافی حاکم بر توسعه و استفاده از سیستم‌های نرم‌افزاری، بخشی از ضعف‌هایی است که بر مهندسی نرم‌افزار امنیتی تأثیرگذار است

رویکردهای جامعی در مهندسی الزامات امنیت نرم افزار توصیه شده است، یعنی STORE [7] که مخفف عبارت Security Threat Oriented Requirements Engineering Methodology، SQUARE که مخفف Security QUAlity Requirements Engineering [20] و MSRE (Metamodel of Security Requirements Engineering) [24] است.

۴. معماری و طراحی امنیتی

در توسعه نرم افزار، معماری به عنوان طراحی در نظر گرفته می شود، اما همه طراحی ها یک معماری در نظر گرفته نمی شوند [۱۵]. معماری تصمیمات مهم طراحی را نشان می دهد که یک سیستم را تشکیل می دهند، جایی که اهمیت با هزینه تغییر اندازه گیری می شود [۴]. معماری و طراحی نرم افزار باید ابهامات را برطرف کرده و به سوالات مهم پاسخ دهد. فاز معماری و طراحی مربوط به پاسخ به این سوال است که "چگونه". ایده های روشن ترجمه شده و به واقعیت تبدیل می شوند. با نگاهی به معماری و طراحی از منظر عملکردی، این تبدیل از ایده به واقعی برای کیفیت کلی و موفقیت نرم افزار ارائه شده بسیار ضروری است. در توسعه سیستم های نرم افزاری ایمن، معماری و طراحی مهمترین مرحله SDLC در نظر گرفته می شود. تصمیمات خوب اتخاذ شده در این مرحله نه تنها رویکرد و ساختاری را ایجاد می کند که در برابر حمله انعطاف پذیرتر و مقاوم تر است، بلکه اغلب به تجویز و راهنمایی تصمیمات خوب در مراحل بعدی مانند کدگذاری و آزمایش کمک می کند. تصمیمات بد اتخاذ شده در این مرحله می تواند منجر به نقص های طراحی شود که حتی با هوشمندانه ترین و منظم ترین کدها و تلاش های آزمایشی هرگز نمی توان بر آنها غلبه کرد یا آنها را حل کرد [۸].

۳. مدیریت ریسک در SDLC ایمن

اهداف امنیتی خاص معماری و طراحی نرم افزار عبارتند از: معماری امنیتی عملکردی جامع (ویژگی ها و قابلیت های امنیتی به طور کامل فعال هستند)، مقاومت در برابر حمله (حاوی حداقل ضعف های امنیتی است که می توان از آنها سوء استفاده کرد)، تحمل حمله (در حین مقاومت در برابر حمله، عملکرد نرم افزار و قابلیت ها تحت تأثیر بی‌دلیل قرار نگرفته است)، انعطاف‌پذیری حمله (در صورت حمله موفق، تأثیرات روی نرم‌افزار به حداقل می‌رسد).

تکنیک ها و فعالیت های مختلفی وجود دارد که می توان در این مرحله انجام داد. اینها مدل‌سازی تهدید و ریسک، طراحی برای برآوردن الزامات امنیتی، الگوهای طراحی امنیتی، دستورالعمل‌های طراحی برای توسعه نرم‌افزار ایمن، طراحی امنیتی و چک لیست معماری، و طراحی امنیتی و بررسی معماری هستند. برای اطلاعات بیشتر در مدیریت ریسک در Secure SDLC به ۳ مراجعه کنید.

مدل‌سازی تهدید [۲۶] شامل تعریف سطح حمله نرم‌افزار با کاوش در عملکرد آن برای مرزهای اعتماد، نقاط ورودی، جریان داده‌ها و نقاط خروج است. مدل‌های تهدید معمولاً زمانی انجام می‌شوند که الزامات عملکردی کامل شوند. مدل‌سازی تهدید تضمین می‌کند که طراحی مکمل اهداف امنیتی است که تصمیم‌های مبادله و اولویت‌بندی تلاش‌ها و کاهش خطر مسائل امنیتی در طول توسعه و عملیات را می‌گیرد. یکی از موثرترین فعالیت های انجام شده در این مرحله مدل سازی تهدید است. مدل‌سازی ریسک، ریسک‌ها، رتبه آنها و کاهش‌ها را مورد بحث قرار می‌دهد. تهدیدهای رتبه‌بندی معمولاً با نگاه کردن به اهداف تجاری سازمان، الزامات انطباق و نظارتی و مواجهه‌های امنیتی به دست می‌آیند. یک مدل تهدید به عنوان یک مشخصات در نظر گرفته می‌شود - درست مانند یک مشخصات عملکردی - که معماری مورد نیاز برای اجرای مشخصات عملکردی را تعریف می‌کند و آزمایشی که نحوه برنامه‌ریزی شما برای اطمینان از اینکه طرح اجرا شده مطابق با الزامات مشخصات عملکردی است را مشخص می‌کند. درست مانند سایر مشخصات، یک مدل تهدید یک سند زنده است - همانطور که طراحی را تغییر می دهید، باید به عقب برگردید و مدل تهدید خود را به روز کنید تا ببینید آیا تهدید جدیدی ظاهر می شود یا خیر. مدل‌سازی تهدید موثرترین روش مدیریت ریسک در فاز معماری و طراحی در نظر گرفته می‌شود [۲۶]، اما همچنین هزینه‌ترین روش است [۲۵].

بیشتر آسیب‌پذیری‌های امنیتی نرم‌افزار به‌جای مسائل مربوط به کدنویسی، مسائل طراحی هستند. مسائل امنیتی طراحی مانند نقص‌های منطق تجاری را نمی‌توان در کد شناسایی کرد و باید با مدل‌سازی تهدید و مدل‌سازی مورد سوء استفاده در مرحله طراحی بررسی شود. مدل‌سازی تهدید

طبیعتاً یک تلاش تکراری است که برای شناسایی تهدیدها استفاده می‌شود. با شناسایی اهداف امنیتی نرم افزار همانطور که در الزامات امنیتی توضیح داده شده است شروع می شود. مدل‌سازی تهدید، نرم‌افزار را به ساختارهای فیزیکی و منطقی تقسیم می‌کند، مصنوعات نرم‌افزاری را تولید می‌کند که شامل نمودارهای جریان داده، سناریوهای استقرار انتها به انتها، نقاط ورود و خروج مستند، پروتکل‌ها، مؤلفه‌ها، هویت‌ها و خدمات است.

۵. اجرای ایمن

امنیت نرم افزار چهار رکن از نظر طراحی ایمن، به طور پیش فرض ایمن، با پیاده سازی ایمن و از طریق ارتباط امن است. Secure By Design را می توان با ایجاد مرزهای اعتماد، عدم اختراع مجدد چرخ، ایجاد اقتصاد مکانیزم، اعمال بی میلی اعتماد، دستیابی به طراحی باز، به حداقل رساندن سطح حمله و ایمن کردن ضعیف ترین پیوند به دست آورد. ایمن به صورت پیش‌فرض با استفاده از حداقل امتیاز، استفاده از رد پیش‌فرض و شکست ایمن قابل دستیابی است. ایمن با پیاده سازی را می توان با مقبولیت روانشناختی، بکارگیری کمترین مکانیسم رایج، اعتبارسنجی ورودی ها، ایمن سازی داده ها در حالت استراحت، جلوگیری از حملات بای پس، ممیزی و تایید، و اعمال دفاع عمیق به دست آورد. امنیت از طریق ارتباط را می توان با ایمن سازی روابط اعتماد و به کارگیری شیوه های خوب ارتباط ایمن به دست آورد.

گری مک گراو اظهار داشت که پیچیدگی به عنوان دشمن امنیت در نظر گرفته می شود [۱۸]. هر نرم افزار دارای سطح و ساختار متفاوتی از پیچیدگی است [۶]. پیچیدگی در ساختار نرم افزار که به دلیل خوشه ای از عناصر برنامه موجود در یک برنامه یافت می شود شامل موارد زیر است:

پیچیدگی معماری به دلیل تعداد زیاد زیرسیستم های به هم پیوسته به وجود می آید.

پیچیدگی چرخه ای به دلیل تعداد زیاد حلقه ها و دستورات if ایجاد می شود.

• پیچیدگی شناختی را می توان با مشاهده رفتار کاربران در یافتن راه حل برای یک کار یا مشکل خاص کشف کرد. تمرکز اصلی آن بر جنبه قابل درک بودن کد است.. Open Web Application Security Project (OWASP) 10 مهم ترین خطر امنیتی برنامه های وب را منتشر کرد.

پیچیدگی در ورودی‌ها همچنین می‌تواند عاملی برای ایجاد آسیب‌پذیری در سیستم و خطرات امنیتی بزرگ مانند اعمال در سیستم عامل یا صفحات به مرورگرهای وب باشد.

برای شروع توسعه نرم‌افزار ایمن، باید شیوه‌های کدگذاری ناامن و نحوه استفاده از آنها را بدانید [۲۱]. طراحی های ناامن می تواند منجر به "خطاهای عمدی" شود، یعنی کد به درستی پیاده سازی شده است اما نرم افزار به دست آمده دارای یک آسیب پذیری امنیتی است. طرح های ایمن نیاز به درک هر دو نوع الزامات دارند. کد نویسی ایمن مستلزم درک ویژگی های پیاده سازی است. تجزیه و تحلیل ریسک همه چیز در مورد دامنه، دارایی ها، تهدیدها و چه اتفاقاتی است. بیشتر جهانی است و اولویت بندی حیاتی است. در حالی که کدگذاری دفاعی یک فرم طراحی دفاعی است که به منظور اطمینان از ادامه عملکرد نرم افزار در شرایط غیرمنتظره است. این ایده را می توان به عنوان کاهش یا حذف خطاهای احتمالی در نظر گرفت. تکنیک های برنامه نویسی دفاعی به خصوص زمانی که ممکن است از یک نرم افزار سوء استفاده شود استفاده می شود. در صورت داشتن یک تغییر کوچک در کد که منجر به تغییر بزرگ در تجزیه و تحلیل ریسک می شود پروژه امنیت برنامه وب باز [1] (OWASP)را نشان می دهد که ۱۰ خطر مهم امنیتی برنامه وب را منتشر کرد.

طیف گسترده‌ای از پلتفرم‌ها، چارچوب‌ها و ابزارهای توسعه وجود دارد که برخی از جنبه‌های کدنویسی ایمن را پوشش می‌دهند، اما کنترل‌های امنیتی ضروری اغلب مفقود، ناقص یا به سادگی اشتباه هستند. این شکاف‌ها گاهی اوقات توسعه‌دهندگان را مجبور به طراحی و ساخت مکانیسم‌های امنیتی خانگی می‌کنند که منجر به اتلاف وقت و برخی مشکلات امنیتی و باگ‌ها می‌شود. استفاده از کنترل های امنیتی با ساختن آنها کاملاً متفاوت است. بیشتر توسعه دهندگان باید بر روی استفاده از کنترل های امنیتی به جای توسعه آنها تمرکز کنند. Enterprise Security API (ESAPI) به گونه‌ای طراحی شده است که جنبه‌هایی از امنیت برنامه‌ها را به صورت خودکار در بر بگیرد و این مسائل را برای توسعه‌دهندگان شفاف کند. پروژه OWASP Enterprise Security API (ESAPI) به مهندسان کمک می کند تا در برابر نقص های طراحی و پیاده سازی مرتبط با امنیت محافظت کنند. همچنین ۱۰ روش کدگذاری امن از CERT به عنوان توصیه های مستقل از زبان وجود دارد [۲۱].

۶. تست امنیتی

تست نرم افزار را می توان با اجرای یک برنامه یا سیستم با هدف یافتن خطاها تعریف کرد [۲۳]. همچنین می تواند به عنوان ارزیابی قابلیت سیستم و تعیین اینکه الزامات آن برآورده شده است، دیده شود. آزمایش اهداف مختلفی برای بهبود کیفیت، تأیید و اعتبار سنجی و تخمین قابلیت اطمینان به طور معمول، تست نرم افزار در چند فاز تکراری اتفاق می افتد. هر مرحله شامل برخی از فعالیت های تست امنیتی است و در هر مرحله توضیح داده می شود: تست واحد، تست یکپارچه سازی، تست تضمین کیفیت، تست پذیرش کاربر.

تست امنیت برای آزمایش الزامات امنیتی با توجه به ویژگی های امنیتی (به عنوان مثال یکپارچگی، محرمانه بودن، در دسترس بودن، مجوز، احراز هویت، و عدم انکار) است. تست امنیت آزمایش می کند که آیا ویژگی های امنیتی مورد نظر به درستی اجرا شده است یا خیر. مطابقت با ویژگی های امنیتی را نشان می دهد. هدف آزمایش منظم این است که اطمینان حاصل شود که برنامه از نظر ویژگی ها و عملکرد، نیازهای مشتری را برآورده می کند. معمولاً موارد استفاده معمولی و با توجه به الگوهای استفاده رایج مورد انتظار را آزمایش می کند. هدف آزمایش امنیتی این است که اطمینان حاصل شود که برنامه الزامات امنیتی را برآورده می کند که اغلب غیر کاربردی هستند و بیشتر به موارد سوء استفاده علاقه مند هستند. گام های کلیدی برای یک برنامه تست امنیتی شفاف و موفق عبارتند از:

۱. شناسایی محدوده تست امنیتی

۲. تولید و اجرای تست مورد

اهداف اصلی مرحله اول عبارتند از: اعتبارسنجی و تأیید اینکه برنامه‌ها با الزامات امنیتی مطابقت دارند و آسیب‌پذیری‌ها را در محیط داده شده برنامه پیدا می‌کنند. انجام یک ارزیابی امنیتی دقیق از یک برنامه یک کار پیچیده است که باید مانند هر کار تجزیه و تحلیل نرم افزار دیگری با روش شناسی، روش های آزمایش، مجموعه ای از ابزارهای مفید، مهارت ها و دانش به آن نزدیک شد. تست نفوذ خودکار و دستی ممکن است برای کشف آسیب‌پذیری‌های امنیتی حیاتی استفاده شود. هر مرحله از توسعه نرم افزار باید استراتژی تست امنیتی خاص خود را داشته باشد.

امنیت یک برنامه با تلاش برای نقض کنترل های امنیتی داخلی آزمایش می شود [۱۱]. این تکنیک تضمین می کند که مکانیسم های حفاظتی در سیستم به اندازه کافی برای ایمن کردن برنامه از دسترسی نامناسب و غیرمجاز کافی هستند. تستر سیستم را با درخواست های مداوم بارگذاری می کند و در نتیجه سرویس به دیگران را رد می کند. آزمایش‌کننده ممکن است عمداً باعث شود که خطاهای سیستم باعث نقض امنیت در هنگام بازیابی شود یا ممکن است در داده‌های ناامن جستجو کند تا کلید ورود به سیستم را پیدا کند. حوزه‌های زیر باید از نظر امنیت، کنترل‌های دسترسی، احراز هویت، مدیریت رمز عبور، اعتبارسنجی ورودی، مدیریت استثنا، ذخیره‌سازی و انتقال امن داده‌ها، ثبت و نظارت و هشدار، مدیریت تغییرات، و ارزیابی‌ها و ممیزی‌های امنیتی دوره‌ای آزمایش شوند.

سرریز بافر، تزریق SQL، اسکریپت بین سایتی، دستکاری پارامترها، مسمومیت کوکی، فیلدهای مخفی، گزینه های اشکال زدایی، ورودی نامعتبر، مجوز خراب، احراز هویت شکسته و مدیریت جلسه، برخی از حوزه هایی هستند که موارد تست باید برای امنیت ایجاد شوند. آزمایش کردن. در حالت ایده آل، تست امنیتی باید در پایان تست یکپارچه سازی عملکردی و تست عملکرد انجام شود [۱۱]. این به شناسایی تهدیدات امنیتی پنهان در برنامه کمک می کند. پس از تکمیل تست امنیتی، یافته ها باید در یک گزارش خلاصه شود. گزارش خلاصه باید حاوی جزئیاتی مانند انواع آزمایش‌های انجام‌شده و خطرات امنیتی شناسایی‌شده، همراه با رتبه‌بندی باشد که به کسب‌وکار کمک می‌کند تا در مورد استقرار برنامه تصمیم‌گیری کند. توسعه‌دهندگان همچنین می‌توانند با استفاده از تکنیک‌های فازی، آزمایش‌های امنیتی مستقیم را انجام دهند. Fuzzing روشی برای کشف عیوب در نرم افزار با ارائه ورودی غیرمنتظره و نظارت بر استثناها است. Fuzzing، به ساده‌ترین عبارت، ارسال داده‌های تصادفی به رابط‌های برنامه کاربردی ((APIاست که برنامه به آنها متکی است و تعیین اینکه آیا، چه زمانی و چگونه ممکن است نرم‌افزار را خراب کند.

تست مبتنی بر ریسک یکی از تکنیک های رایج تست امنیت است [۱۲]. خطرات و احتمال و تأثیر آنها تلاش های امنیتی را هدایت می کند. تلاش آزمون بر اساس امتیاز ریسک اولویت بندی می شود. ابتدا نواحی پرخطر، سپس با خطر متوسط و در نهایت نواحی کم خطر آزمایش می شوند. هدف اصلی رسیدن به سطح قابل قبولی از ریسک است. در تلاش های دیگر برای آزمایش امنیت یک سیستم نرم افزاری،بررسی کد منبع دستی زمانی شروع می شود که کد کافی برای بازبینی وجود داشته باشد. بررسی کد منبع دستی معمولاً بر یافتن آسیب‌پذیری‌های امنیتی در سطح کد متمرکز است. بررسی کد منبع دستی برای هیچ نقص امنیتی که می تواند بر کیفیت کلی کد تأثیر بگذارد نیز مفید است. گزارش‌های اشکال اغلب حاوی یک توصیه اصلاحی خاص توسط بازبین است تا توسعه‌دهنده بتواند آن را به درستی برطرف کند. بررسی‌های کد دستی گران هستند زیرا شامل تلاش‌های دستی زیادی می‌شوند و اغلب متخصصان امنیتی را برای کمک به بررسی درگیر می‌کنند. با این حال، بررسی های

دستی ارزش خود را بارها و بارها در مورد دقت و کیفیت ثابت کرده اند. آنها همچنین به شناسایی آسیب‌پذیری‌های منطقی کمک می‌کنند که معمولاً توسط تحلیلگرهای کد استاتیک خودکار قابل شناسایی نیستند.

شرکت‌های متوسط تا بزرگ نمی‌توانند هر بار یک بازبینی کد دستی روی هر برنامه را انجام دهند. در عوض، بسیاری برای کمک به تحلیلگرهای کد منبع خودکار متکی هستند. اولویت‌های معمول توسعه نرم‌افزار عبارتند از زمان‌بندی، هزینه، ویژگی‌ها و سپس کیفیت - در اکثر موارد، به ترتیب. فشار از دیدگاه زمان به بازار می تواند بر کیفیت و انعطاف پذیری نرم افزار تأثیر منفی بگذارد و گاهی اوقات باعث به تعویق افتادن افزودن ویژگی ها به نرم افزار می شود. سازمان‌هایی که دارای فرآیند SDLC بالغ هستند، معمولاً به دلیل کیفیت نرم‌افزار و الزامات انعطاف‌پذیری، با هزینه‌های اضافی کمی روبرو هستند و صرفه‌جویی در هزینه‌های مربوط به بهبود فرآیند بسیار بیشتر از هزینه فعالیت‌های توسعه‌دهنده اضافه شده است. آنها طیف گسترده ای از نماها/گزارش ها و روندها را در مورد وضعیت امنیتی پایه کد ارائه می دهند و می توانند به عنوان مکانیزمی موثر برای جمع آوری معیارهایی استفاده شوند که نشان دهنده پیشرفت و بلوغ فعالیت های امنیتی نرم افزار است. تحلیلگرهای کد منبع در بازه‌های زمانی شگفت‌انگیز سریعی کار می‌کنند که اگر به صورت دستی انجام می‌شد، چندین هزار نفر ساعت طول می‌کشید.

اگرچه تحلیلگرهای کد منبع خودکار در عملکرد با هزینه های افزایشی کم قوی هستند، در گرفتن میوه های معمولی کم آویزان خوب هستند، توانایی مقیاس دهی به چندین هزار خط کد را دارند، و در انجام سریع کارهای تکراری خوب هستند، اما همچنین می توانند به خوبی کار کنند. چند اشکال ابزارهای خودکار تمایل دارند تعداد بالایی از موارد مثبت کاذب را گزارش کنند. گاهی اوقات چندین ماه طول می کشد تا یک سازمان ابزار را برای کاهش این موارد مثبت کاذب تنظیم کند، اما سطحی از نویز همیشه در یافته ها باقی می ماند. تحلیلگرهای کد منبع در تشخیص نقص های منطق تجاری ضعیف هستند. برخی از انواع دیگر حملاتی که تجزیه و تحلیل خودکار نمی تواند آنها را شناسایی کند عبارتند از نشت اطلاعات پیچیده، نقص های طراحی، آسیب پذیری های ذهنی مانند جعل درخواست بین سایتی، شرایط مسابقه پیچیده و حملات چند مرحله ای.

یک شناخته شده تست امنیتی جعبه سیاه، تست نفوذ است. در یک تست نفوذ، یک برنامه یا سیستم از بیرون در راه اندازی آزمایش می شود که با حمله واقعی یک شخص ثالث مخرب قابل مقایسه است. تست نفوذ یک تست امنیتی است که در آن ارزیابان تلاش می کنند تا ویژگی های امنیتی یک سیستم را بر اساس درک خود از طراحی و پیاده سازی سیستم دور بزنند. تعیین میزان آسیب پذیری شبکه یک سازمان و میزان آسیبی که در صورت به خطر افتادن شبکه ممکن است رخ دهد بسیار مهم است. یک تست نفوذ می تواند حملات داخلی و خارجی را شبیه سازی کند. تست نفوذ نه تنها برای یافتن آسیب‌پذیری‌های احتمالی، بلکه برای یافتن قابلیت بهره‌برداری حمله و میزان تأثیر تجاری در صورت بهره‌برداری موفق انجام می‌شود.

۷. مدل هایی برای توسعه نرم افزار ایمن

این مدل ها را می توان به طور کلی به دو دسته مدل های تجویزی و توصیفی دسته بندی کرد. مدل‌های تجویزی راهنمایی در مورد کارهایی که افراد باید انجام دهند ارائه می‌کنند، در حالی که مدل‌های توصیفی برای توسعه نرم‌افزار امنیتی، مقایسه‌ای از آنچه یک شرکت در مقایسه با سایرین انجام می‌دهد، ارائه می‌کند. مدل های توصیفی به عنوان معیار با سایر شرکت ها استفاده می شود. مدل‌های تجویزی مانند چرخه حیات توسعه نرم‌افزار مایکروسافت، فرآیند امنیت نرم‌افزار جامع، سبک وزن [CLASP]، توسعه، امنیت و عملیات (DevSecOps۵) و مدل‌های توصیفی مانند مدل بلوغ تضمین نرم‌افزار باز [2] (OpenSAMM)و امنیت ساختمان در مدل سررسید Microsoft SDL .[19] (BSIMM) شیوه های امنیتی را توزیع می کندبه مراحل مختلف توسعه نرم افزار SDLC. CLASP یک عنصر فرآیند از پیش تعریف شده است که ممکن است به هر فرآیند توسعه نرم افزاری تزریق شود در حالی که [22] DevSecOpsدارای پنج مرحله اصلی است، برنامه ریزی برای امنیت، تعامل با توسعه دهندگان و درگیر شدن، مسلح کردن توسعه دهندگان، خودکار کردن فرآیند، و استفاده هوشمندانه از ابزارهای قدیمی. .

در میان مدل‌های توصیفی، مدل امنیت ساختمان در بلوغ ((BSIMMتحریک تغییرات فرهنگی در ایجاد نرم‌افزار امن را تشویق می‌کند. هدف اصلی این پروژه ایجاد یک مدل بلوغ با توجه به جمع آوری داده های واقعی از ۹ طرح توسعه نرم افزار در مقیاس بزرگ است. OpenSAMM یک مدل بلوغ کاملاً تعریف شده (با ابزارهای خود ارزیابی و برنامه ریزی) برای توسعه و استقرار نرم افزار ایمن را نشان می دهد۵. مدل DevSecOps

۸. نتیجه گیری

بسیاری از افراد و موسسات منابع خود را در محصولات نرم افزاری مختلف قرار داده اند. توسعه نرم‌افزار ایمن یک ضرورت است که می‌توان با پیاده‌سازی اصول مورد بحث توسعه نرم‌افزار ایمن و همچنین مراقب مقابله با روش‌های جدید نقض امنیت نرم‌افزار به آن دست یافت. این راهنما به وضوح نشان داده است که توسعه نرم افزار ایمن زمینه بسیار دشواری نیست که فقط به متخصصان فناوری اطلاعات (فناوری اطلاعات) واگذار شود. این راهنما ارائه آسان و قابل فهم امنیت سیستم های نرم افزاری را به روشی ساده ارائه کرده است که هر ذینفع را قادر می سازد آن اقدامات اساسی اما حیاتی را انجام دهد. محققان مبنایی برای توسعه اطلاعات بیشتر بر اساس یافته های کار ارائه شده دارند. اطلاعات، استراتژی‌ها، اصول و مدل‌های موجود را که می‌توانند در توسعه نرم‌افزار امن مورد استفاده قرار گیرند، شناسایی کرده است. بر اساس اثربخشی هر یک از اصول، استراتژی و مدل ارائه شده می توان کارهای تحقیقاتی بیشتری را انجام داد.

منابع

[۱] Open Web Application Security Project (OWASP). (دسترسی در ۱۲ فوریه ۲۰۲۰)، ۲۰۲۰ https://owasp.org/پروژه.

[۲] OWASP SAMM. (دسترسی در ۱۲ فوریه ۲۰۲۰)، ۲۰۲۰ https:// wiki.owasp.org/index.php/OWASP_SAMM_Project.

[۳] ثامر الحامد و ممدوح النزی. مدیریت تداوم کسب و کار و قابلیت های بازیابی بلایا در عربستان سعودی ممدوح آلنزی، آلکا آگراوال، راجیو کومار و رئیس احمد خان. ارزیابی عملکرد امنیت برنامه های کاربردی وب از طریق یک رویکرد تصمیم گیری چند معیاره ترکیبی مبتنی بر فازی: دیدگاه تاکتیک های طراحی دسترسی IEEE، ۲۵۵۴۳:۲۵۵۶۸–، ۲۰۲۰. [۵] ممدوح النظی و صدیق المعیرفی. خطرات امنیتی در چرخه عمر توسعه نرم افزار مجله بین المللی فناوری و مهندسی اخیر (IJRTE)، ۸ (13)، ۲۰۱۹.

[۶] ممدوح النزی و محمد زرور. در مورد رابطه بین پیچیدگی نرم افزار و امنیت مجله بین المللی مهندسی نرم افزار و برنامه های کاربردی (IJSEA)، ۱۱ (1)، ۲۰۲۰.

[۷] دکتر تاریک جمال انصاری، دیرندرا پاندی، و ممدوح آلنزی. فروشگاه: متدولوژی مهندسی الزامات تهدید گرا. مجله دانشگاه ملک سعود - علوم کامپیوتر و اطلاعات، ۲۰۱۸.

[۸] ایوان آرس، کاتلین کلارک فیشر، نیل داسوانی، جیم دلگروسو، دنی دیلون، کریستف کرن، تادیوشی کوهنو، کارل لندور، گری مک‌گرو، بروک شوئنفیلد، و دیگران. اجتناب از ۱۰ نقص اصلی طراحی امنیتی نرم افزار. IEEE Computer Society Center for Secure Design (CSD)، .Techنماینده، ۲۰۱۴.

[۹] برد آرکین، اسکات استندر، و گری مک‌گرو. تست نفوذ نرم افزار امنیت و حریم خصوصی IEEE، ۳ 87-84 :(1)، ۲۰۰۵.

[۱۰] الکساندر بارتل، ژاک کلاین، ایو لو ترائون، و مارتین مونپروس. ایمن سازی خودکار نرم افزار مبتنی بر مجوز با کاهش سطح حمله: برنامه ای برای اندروید. در سال ۲۰۱۲ مجموعه مقالات بیست و هفتمین کنفرانس بین المللی IEEE/ACM در زمینه مهندسی نرم افزار خودکار، صفحات ۲۷۴-۲۷۷. IEEE، ۲۰۱۲.

[۱۱] مایکل فلدرر، ماتیاس بوچلر، مارتین جانز، آخیم دی بروکر، روث برو، و الکساندر پرتشنر. تست امنیت: یک نظرسنجی در Advances in Computers، جلد ۱۰۱، صفحات ۱-۵۱. الزویر، ۲۰۱۶.

[۱۲] مایکل فلدرر و اینا شیفردکر. طبقه بندی تست های مبتنی بر ریسک مجله بین المللی ابزارهای نرم افزاری برای انتقال فناوری (STTT)، ۱۶ (5): 559-568، ۲۰۱۴.

[۱۳] لین فوچر و روسوو فون سولمز. دستورالعمل های توسعه نرم افزار ایمن در مجموعه مقالات کنفرانس تحقیقاتی سالانه ۲۰۰۸ مؤسسه دانشمندان رایانه و فناوری اطلاعات آفریقای جنوبی در مورد تحقیقات فناوری اطلاعات در کشورهای در حال توسعه: سوار بر موج فناوری، صفحات ۵۶-۶۵، ۲۰۰۸.

[۱۴] یوهان گرگویر، کوئن بوینز، بارت دی وین، ریکاردو اسکانداریاتو و ووتر جوسن. در مورد فرآیند توسعه نرم افزار ایمن: مقایسه clasp و .sdl در سومین کارگاه بین المللی مهندسی نرم افزار برای سیستم های امن ((SESS'07: ICSE Workshops 2007، صفحات ۱-۱. IEEE، ۲۰۰۷.

[۱۵] نافیل کازی، دیپا پرسار، و فوزیه صدیقی. راهنمای طراحی امنیتی سایبرنومیکس، ۱ (۶): ۳۷-۴۰، ۲۰۱۹.

[۱۶] مایک منیون و بری کیپنس. الزامات هوشمند. یادداشت های مهندسی نرم افزار، ۲۰ (۲): ۴۲-۴۷، ۱۹۹۵.

[۱۷] جان مک درموت و کریس فاکس. استفاده از مدل های مورد سوء استفاده برای تجزیه و تحلیل الزامات امنیتی در مجموعه مقالات پانزدهمین کنفرانس سالانه برنامه های کاربردی امنیت رایانه ((ACSAC'99، صفحات ۵۵-۶۴. IEEE، ۱۹۹۹.

[۱۸] گری مک گرو. امنیت نرم افزار: امنیت ساختمان در. ادیسون-وسلی حرفه ای، ۲۰۰۶.

[۱۹] گری مک گراو و برایان شطرنج. مدل امنیت ساختمان در سررسید (.(BSIMMدر هجدهمین سمپوزیوم امنیت USENIX (USENIX Security '09)، مونترال، کبک، آگوست ۲۰۰۹. انجمن .USENIX

[۲۰] نانسی آر مید و تد استهنی. روش شناسی مهندسی الزامات کیفیت امنیت (مربع). یادداشت های مهندسی نرم افزار ACM SIGSOFT، ۳۰ (4): 1-7، ۲۰۰۵.

[۲۱] مارک اس مرکو و لاکشمیکانت راگاوان. توسعه نرم افزار ایمن و انعطاف پذیر. انتشارات اورباخ، ۲۰۱۰.

[۲۲] هاوارد میرباکن و ریکاردو کولومو-پالاسیوس. :Devsecopsبررسی ادبیات چند آوایی در کنفرانس بین المللی بهبود فرآیند نرم افزار و تعیین قابلیت، صفحات ۱۷-۲۹. اسپرینگر، ۲۰۱۷.

[۲۳] بروس پاتر و گری مک گرو. تست امنیت نرم افزار امنیت و حریم خصوصی IEEE، ۲۵۲-81 :(5)، ۲۰۰۴.

[۲۴] سون تورپ. مشکل با الزامات امنیتی در ۲۰۱۷ (IEEE 25th International Requirements Engineering Conference (RE، صفحات ۱۲۲-۱۳۳. IEEE، ۲۰۱۷.

[۲۵] الین ونسون، شیائومنگ گو، زیدی یان، و بری بوهم. توسعه نرم افزار ایمن هزینه یابی: یک مطالعه نقشه برداری سیستماتیک در مجموعه مقالات چهاردهمین کنفرانس بین المللی در دسترس بودن، قابلیت اطمینان و امنیت، صفحات ۱ تا ۱۱، ۲۰۱۹.[۲۶] ونجون شیونگ و رابرت لاگرستروم. مدل‌سازی تهدید - مروری بر ادبیات سیستماتیک کامپیوتر و امنیت، ۲۰۱۹.